Multiclass Classification

Intro to Machine Learning: Beginner Track

Feedback form: bit.ly/btrack-w22-feedback
Discord: bit.ly/ACMdiscord

acm.ai

Logistic Regression (Binary Classification) Review

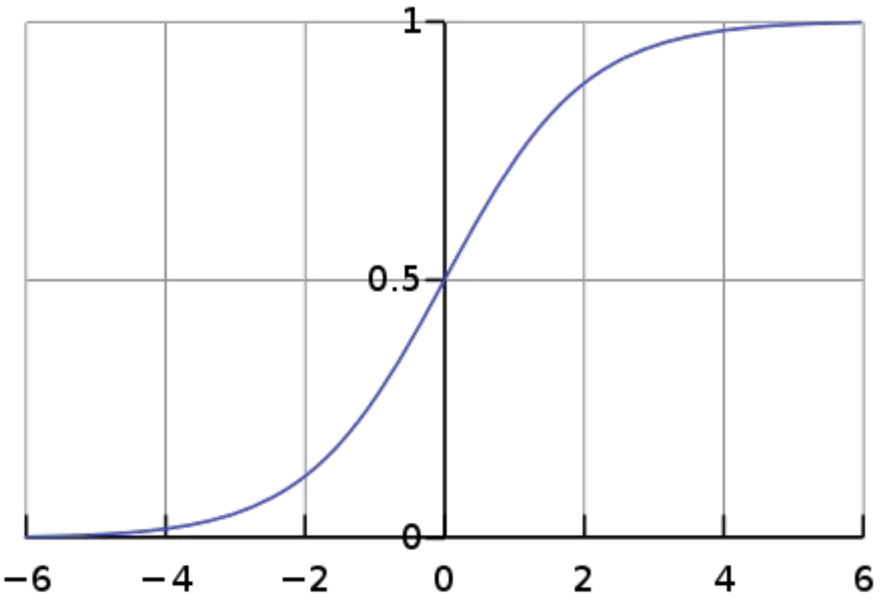
The Objective

- Given the **input features** of an object, find a **model** that predicts the **probability** that the object belongs to class **1**
  - Aside : How do we get the probability of object belonging to class 0 from this?
- We need a **hypothesis** to find this probability
  - What is a hypothesis?
- For these kinds of **classification problems,** one potential hypothesis is the **logistic regression model**

Sigmoid Function

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

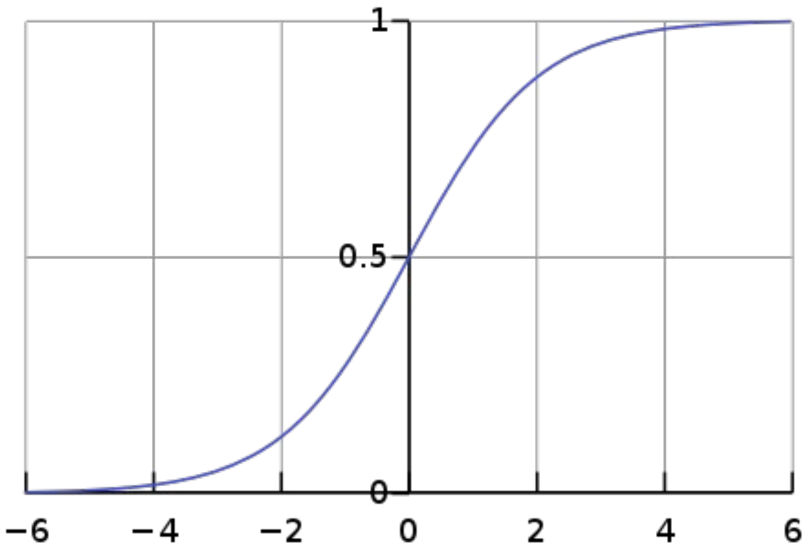
- If x is negative, sigma(x) < 0.5
- If x is positive, sigma(x) > 0.5
- Also, 0 < sigma(x) < 1 over its entire domain
- Ideal for modeling probability distribution



Logistic Regression

$$\hat{y}(x) = \frac{1}{1 + e^{-(X \cdot W + b)}}$$

- An input needs to be classified as **0** or **1**
- We need to find the **decision boundary** or the **W** and **b** for our model
- This is known as **binary** classification



acm acm.ai

Probability of being a particular class

- Think of the output of the model as the **probability** of the input being class 1 given the features X.

$$\hat{y}(x) = P(Y = 1|X)$$

- This is read as “Probability that the label **Y** is **1** given the features **X** we have”

Cost Function: Binary Cross-Entropy Loss

$$L(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

- **ŷ** : prediction
- **y** : label

- What happens when **y** is **1**? $L(\hat{y}, y) = -\log(\hat{y})$
- What happens when **y** is **0**? $L(\hat{y}, y) = -\log(1 - \hat{y})$

Gradient Descent

The derivatives **dL /dw** and **dL / db** are similar to those in linear regression.

$$\frac{\partial L}{\partial w} = \frac{1}{m} X^T (\hat{Y} - Y) \qquad w = w - \alpha \frac{\partial L}{\partial w}$$

$$\frac{\partial L}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} (\hat{Y}_i - Y_i) \qquad b = b - \alpha \frac{\partial L}{\partial b}$$

Yhat is a column vector (m x 1) containing all the predictions, ***Y*** is a column vector (m x 1) with the labels/target, and **X** is the data (m x n)

Check out the full [derivation] if you are interested in the Math
Credit to **towardsdatascience.com**

Quick Poll: Logistic Regression Review

Which task is logistic regression well suited for?

a. Predicting the price of a house
b. Predicting whether to approve a loan or deny a loan
c. Generating pictures of dogs and cats
d. Facial recognition software

Binary Classification Demo!

https://tinyurl.com/btrack-w22-playground

Multiclass Classification

Labels

Imagine that we have a bunch of photos of cats, dogs, chickens, and fish that we want to classify

0

1

2

0

2

3

??

To help our model distinguish them we can assign 0 to cats, 1 to dogs, 2 to chickens, and 3 to fish

Any potential issues with this labeling?

acm.ai

One Hot Encoding

For a single image we can assign a **0 or 1** to each category depending on whether or not the image is under that category

Then we put these labels into a vector indexed by each class

This process is called **one hot encoding**

Cat:
Dog:
Chicken:
Fish:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

One Hot Encoding

Now that our samples have one hot encoded labels, our model needs to have a similarly shaped output so that we can compare our predictions and labels

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Binary Model Review

For **binary** classification and linear regression, a single training example **X** was an n-dimensional **vector** for the n features in the example

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

The weight **W** was an n-dimensional column vector

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix}$$

The bias **b** was a real number

$$b$$

Multi Class Model

For multi-class classification, we have the same input **X**

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

But now our weight **W** is an *(n x c)* matrix where **c** is the number of classes, **n** is the number of features

$$\begin{bmatrix} w_1^1 & w_1^2 & \cdots & w_1^c \\ w_2^1 & w_2^2 & \cdots & w_2^c \\ \vdots & \vdots & \ddots & \vdots \\ w_n^1 & w_n^2 & \cdots & w_n^c \end{bmatrix}$$

Our bias **b** becomes a c-dimensional vector

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_c \end{bmatrix}$$

Multi Class Model

For our animal example, to generate the output we

1) take the pixel values from our image and put them in a vector for our **x**
2) multiply it by our weight matrix and add our bias vector
3) output our prediction **z**

$$\longrightarrow \underset{\substack{(px,\ 1)\\ x}}{\begin{bmatrix} 3 \\ 254 \\ 11 \\ \vdots \\ 19 \end{bmatrix}} \longrightarrow \underset{\substack{(4,\ px)\\ w^T}}{\begin{bmatrix} 8 & 32 & 99 & 10 \\ 13 & 17 & 12 & 38 \\ 14 & 2 & 58 & 31 \\ \vdots & \vdots & \vdots & \vdots \\ 19 & 39 & 13 & 29 \end{bmatrix}^T} \times \underset{\substack{(px,\ 1)\\ x}}{\begin{bmatrix} 3 \\ 254 \\ 11 \\ \vdots \\ 19 \end{bmatrix}} + \underset{\substack{(4,\ 1)\\ b}}{\begin{bmatrix} 2 \\ 3 \\ 19 \\ 8 \end{bmatrix}} \longrightarrow \underset{z}{\begin{bmatrix} 23 \\ 9 \\ 1 \\ 34 \end{bmatrix}}$$

Quick Poll: Challenge Question

In multi-class classification , with ***f*** features, ***c*** classes, and ***m*** training samples for X, the matrix W (weights) will have dimensions:

a. (f, 1)
b. (f, c)
c. (m, f)
d. (c, m)

Softmax

Softmax

$$\mathbf{z} \qquad\qquad\qquad\qquad\qquad\qquad \hat{y}$$

$$\begin{bmatrix} 3 \\ 4 \\ 1 \\ 2 \end{bmatrix} \longrightarrow \frac{e^{z}}{\sum_{i=1}^{c} e^{z_i}} \longrightarrow \begin{bmatrix} 0.24 \\ 0.64 \\ 0.03 \\ 0.09 \end{bmatrix}$$

- takes in the output vector **z** from our model
- outputs vector **ŷ** of probabilities for each class that sums to 1
- Why not use a simple ratio? (Think about negatives!)

Softmax

To convert our outputs **z** to probabilities **ŷ** we,

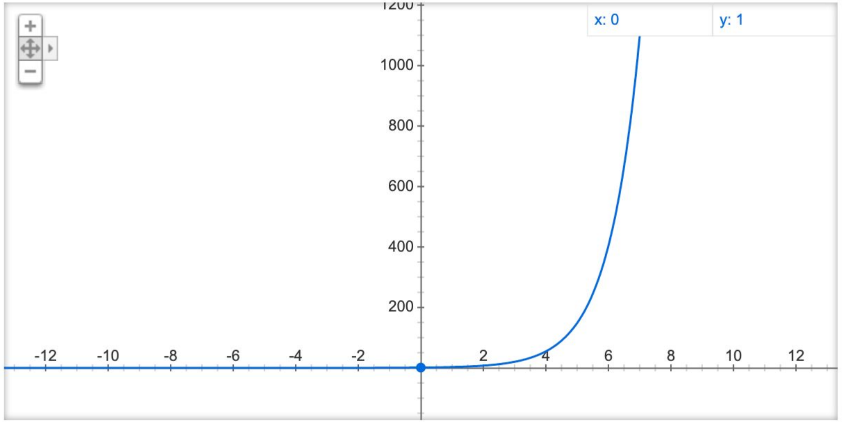
1) raise *e* by component of our output vector **z**
2) divide by the sum of the previous vector to get a vector of probabilities **ŷ**

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_c \end{bmatrix} \longrightarrow \begin{bmatrix} e^{z_1} \\ e^{z_2} \\ e^{z_3} \\ \vdots \\ e^{z_c} \end{bmatrix} \longrightarrow \frac{1}{\sum_{i=1}^{c} e^{z_i}} \begin{bmatrix} e^{z_1} \\ e^{z_2} \\ e^{z_3} \\ \vdots \\ e^{z_c} \end{bmatrix} \longrightarrow \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_c \end{bmatrix}$$

Why use exponential in Softmax?

- exp(x) is always positive
- exp(x) is monotonically increasing
- Empirically works well in multiclass classification

Graph for e^x



More info

Multiclass Cost Function

Cross Entropy

- Need a general loss function that can apply to **c** number of classes
- Needs to have a higher cost if our model makes a bad prediction
  - I.e. the probability for the correct class is far away from 1
- This function is called **cross entropy** or categorical cross entropy

Cross Entropy

$$L(\hat{y}, y) = \sum_{i=1}^{c} -y_i \log(\hat{y}_i)$$

$$\begin{array}{cc} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} .18 \\ .27 \\ .54 \\ .01 \end{bmatrix} \\ \text{y} & \hat{y} \end{array}$$

- The only class that will contribute to the loss is the class that has a **1** in the label
- To minimize the cost, the model needs to make the corresponding class in **ŷ** as close to 1 as possible

Cross Entropy

So total cost across all training sample becomes:

$$J(w,b) = \frac{1}{m}\sum_{j=1}^{m} L(\hat{y}_j, y_j)$$

$$J(w,b) = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{c} -y_{ji} log(\hat{y}_{ji})$$

Gradient Descent in Multiclass Classification

Gradient Descent

The derivatives **dJ /dw** and **dJ / db** are the same as those in binary classification

$$\frac{\partial L}{\partial w} = \frac{1}{m} X^T (\hat{Y} - Y)$$

$$w = w - \alpha \frac{\partial L}{\partial w}$$

$$\frac{\partial L}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} (\hat{Y}_i - Y_i)$$

$$b = b - \alpha \frac{\partial L}{\partial b}$$

Why is this true?
Because **softmax** is a generalization of the **sigmoid** function
and **cross-entropy** loss is a generalization of **binary cross-entropy** loss

Putting it all together

$$\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

$$\longrightarrow \underset{(n, 1)}{\overset{\text{Pixels}}{\begin{bmatrix}3\\254\\11\\\vdots\\19\end{bmatrix}}} \xrightarrow{\text{Linear model}} \underset{(c, 1)}{\overset{\text{Outputs}}{\begin{bmatrix}3\\4\\1\\2\end{bmatrix}}} \xrightarrow{\text{Softmax}} \underset{(c, 1)}{\overset{\text{Probabilities}}{\begin{bmatrix}0.24\\0.64\\0.03\\0.09\end{bmatrix}}} \xrightarrow{\text{Cross Entropy}} 2.4 = \text{loss}$$

Quick Poll

Your model outputs the following probabilities for multi-class classification with 5 classes

[0.3, 0.2, 0.3, 0.1, x]

What is x?

a. 0.3
b. 0.2
c. 0.5
d. 0.1

Multiclass Classification in Code

acm.ai

Binary and Multiclass Classification Demo

- Colab Demo: https://tinyurl.com/btrack-w22-ws5-colab

Thank you! We'll see you next week!

Feedback form: bit.ly/btrack-w22-feedback

Next week: Numpy and Pandas?

Kungfu Panda vs NumPy Pandas. Who would win?

FB group: facebook.com/groups/uclaacmai